High Throughput Analysis of Human B Cell Repertoire Data to Identify Clonal Lineages

By Ishani Kapoor

Senior Honors Thesis

UNC Department of Biostatistics

University of North Carolina at Chapel Hill

April 20th, 2020

Approved:

_____

Dr. Xianming Tan – Thesis Advisor

_____

Dr. Ye Qian – Committee Member

_____

Dr. Jane Monaco – Committee Member

Abstract

B cells are a type of white blood cell which are responsible for producing antibodies. Antibodies are important in human immunity, as they bind to and neutralize foreign invaders. Blood samples were taken from individual patients to sequence the DNA of their B cells. Previous literature points to a variety of ways to analyze immunoglobulin (antibody) sequencing data, however, we focused on using a high throughput methodology to find antibody clones within an individual's B cells. We refer to clones as B cells that are derived from the same parent B cell but have undergone small changes (i.e., mutations) resulting in slight differences. This methodology can be customized by focusing on various parameters to identify clones. Our design focused on three specific parameters for an antibody Heavy (H) chain – having (1) the same Variable gene, Joining gene, and junction length, (2) identical Complementarity Determining Region 3 (CDR3) lengths, and (3) 80% similarity in the CDR3 region. In this project, a single individual's B cell sequencing data was analyzed using R packages to compare these sequences to known germline sequences. An R program was developed to then plot this organized data to create lineage trees, which can help to analyze the path of differentiation a B cell takes and determine mutations occurring within this process. The lineage trees show varying isotypes that were created by the individual's B cells. This summary data can be aggregated with more individual level sequencing data to better understand the isotype switching route, which is important in understanding human antibody development in autoimmune diseases.

Background

B Cells of the Humoral Immune System

B cells and T cells are two types of lymphocytes of the human immune system that function to mount a specific immune response on foreign invaders or antigens[1]. B cells specifically are responsible for producing antibodies, proteins which bind to and neutralize antigens[2]. B cells are formed in the bone marrow and start from a progenitor cell, a precursor cell that gives rise to B cells with different specificity[3]. As shown in Figure 1, there are a variety of steps involved in B cell maturation. These numerous steps provide many opportunities during which a B cell can evolve from its initial precursor cell. Some of these processes include heavy and light chain gene rearrangement and somatic hypermutation[4]. These changes may occur in response to selective pressures to increase the affinity between the antibody and antigen[4]. The sum of the various diversification processes results in multiple sets of clones derived from different ancestor B cells, or collections of B cells that are progenies of a same parental B cell. In addition, B cells can undergo a biological mechanism called isotype switching where the immunoglobulin produced can be changed from one type to another[5]. In this process, the antibody's constant region of the heavy chain is changed, but the variable region does not change[5]. Therefore, antigen specificity is unaffected, and now the antibody can interact with different effector molecules, thereby specifying the functionality of the antibody[5].

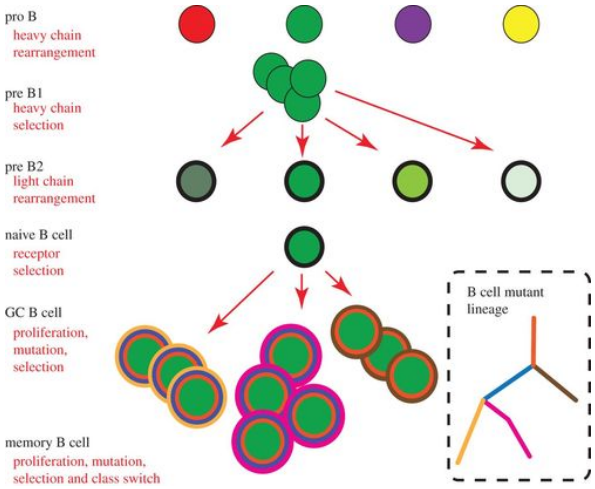


Figure 1. Adapted from Hersheberg, et al., shows the stages of B cell differentiation due to genetic recombination of antibody genes, showing how clones result in the B cell development process[3]. The events in this recombination process are retained in the DNA and can serve as markers to find B cell clones[3].

Sequencing Data

The advent in high-throughput DNA sequencing (HTS) methods has resulted in large amounts of cellular level data[4]. In this study, the investigators obtained blood samples from several subjects and B cells were selected from these samples. For example, patient JLDO, which is used as the example data in later sections, is a 23-year-old female with a 5-month history of a generalized Fogo Selvagem (FS). Each B cell expresses a practically unique B-cell immunoglobulin receptor (BCR)[6]. High throughput screening technology could then be applied to obtain the DNA sequences that corresponds to the unique BCRs of single B cells, with one sequence corresponding to the BCR of one B cell[6]. This sequencing data provides the opportunity to understand the clonal expansion that generates unique antibodies and determine the specific lineage of sequences. The vast number of parameters available to classify clonal lineages leaves room for which parameters to use to define a clone.

One parameter is the Variable (V), Diversity (D), and Joining (J) gene segments, also known as V(D)J recombination[7]. This rearrangement process is specific to lymphocytes and results in breaking and subsequent joining of portions of DNA[7]. The process introduces diversity into the gene segment, allowing for identification of new antigens and production of a diverse set of antibodies throughout the body[7]. The V(D)J recombination process specifically introduces diversity into the complementarity determining region 3 (CDR3) loop[8]. The CDR3 region, combined with CDR1 and CDR2, form the paratope of an antibody where specific antigens can be recognized by this antibody[8]. Therefore, the more recombination introduced diversity in this region, the greater gain in the variety of antigens that can be recognized by antibodies[8]. Using these two parameters, we grouped sequences into clusters according to (1) having the same V gene, J gene, and junction length, (2) having identical CDR3 lengths, and (3) having 80% similarity in the CDR3 region[9].

Research Question and Importance

The advent in high-throughput DNA sequencing methods has resulted in large amounts of data that provide the opportunity to understand the clonal expansion that generates unique antibodies[4]. The vast number of parameters available to classify clonal lineages leaves a lot of room for a variety of combination of parameters. Many experimental methodologies have been investigated to analyze the heavy and light chains of an antibody, including techniques such as hybridoma panels, antibody phage display, single cell cloning, and bulk sequencing of heavy and light chains[3].

Using various packages based in the R programming language and the National Center for Biotechnology Information's (NCBI) IgBLAST tool, an in depth analysis was conducted of the data analysis method of identifying clonal lineages using bulk sequencing of heavy and light

chains[10]. We have tested an easy to follow sequence of steps that identifies clone lineages of DNA sequencing data from a population. This stepwise methodology involves identifying sequences with common germline V and J genes and similarity of length and sequences for the CDR3 region. Understanding clonal evolution of B cells is vitally important to understanding how autoantibodies develop in autoimmune disorders[4,9].

Methodology Review on Clonal Identification

Methods for analyzing immunoglobulin genes and their common ancestor have been of interest for many years, and the majority of analysis methods involve fitting a lineage tree to the observed distances in the data[10]. A 1984 paper by Felsenstein describes two interpretations of the branch lengths[10].

The first is the path length interpretation, which requires calculating observed distance from ancestors to the tips of a lineage tree[10]. A branch length represents the evolutionary distance from ancestor to descendant and can be determined if a set of imaginary ancestors fitting the observed distance are fit, to allow for lineage interpretations[10].

The second interpretation uses a statistical framework, where the assumption is that the distances are independently drawn from a distribution[10]. Here, the branch length is a parameter in a statistical model, estimated using the observed distances[10]. If this distribution is assumed to be normal, as shown in equation 1, where d_{ij} is the observed distance, d'_{ij} is the sum of the lengths of the branches between pairs, and v_{ij} is the variance of the observed distances[10].

Equation 1: $d_{ij} = N{\sim}(d_{ij}, v_{ij})$

This interpretation would then focus on choosing branch lengths to minimize the sum of squares, as shown in Equation 2[10].

Equation 2: $S = \sum_i \sum_j \frac{(d_{ij} - d'_{ij})^2}{v_{ij}}$

Given the development of newer techniques that make it possible to analyze antibodies from individual B cells, statistical methods have been refined over time to study clonal lineages. Kepler presents a method to understand the uncertainties present in reconstructing lineages,[11]. Kepler analyzed heavy and light chain data from individual B cells to present a model containing two features: (1) using all the information available in a set of clonally related immunoglobulin genes and (2) having systematic uncertainty estimates on the non-mutated ancestors[11]. The first step is to estimate a posterior probability, using Bayes theorem (Equation 3), on a common ancestor, α, given a query set of observed immunoglobulin V-gene sequences, Q[11].

Equation 3: $$P(\alpha|Q) = \frac{P(Q|\alpha)P_0(\alpha)}{\sum_{\alpha'} P(Q|\alpha')P_0(\alpha')}$$

Next, a likelihood function is used to describe the probability that the query set arose from a given ancestor by somatic hypermutation[11].

The goal of Kepler's analysis is to make inferences about the clonal history for immunoglobulin genes by computing and utilizing posterior distributions on the selected parameters used for analysis. Additionally, understanding where the uncertainty in assumptions comes from in this method allows for finding the non-mutated ancestor with greater precision[11].

Various distance metrics and linkage methods that can be used to group clonally related sequences were compared in a recent study[12]. The study focused on parameters used in hierarchical clustering of clonal groups, which involves using a distance measure to compare pairs of sequences and a linkage method to measure the distance between groups of sequences[12]. Hierarchical clustering uses these parameters to define cluster and produce a tree relationship between the input sequences[12].

The authors used the following analysis sequence to compare parameters: (1) separate sequences by V, D, J genes, and junction length, (2) assemble sequences into a hierarchy as

defined by the chosen distance metric and linkage method, and (3) partition this hierarchy into distinct clones based on a fixed threshold for distance[12]. This study found that single linkage hierarchical clustering and length normalizing nucleotide Hamming distance are an optimal combination of parameters to group immunoglobulin genes by clonal relatedness[12]. Additionally, this study automated choosing the threshold level for clonal distance[12]. The analysis framework we tested utilized a similar series of steps outlined in this paper, including starting the analysis by looking for the same V, J genes, and junction length. The Hamming distance was also used as our distance metric, however, we continued to use a fixed threshold of 80% similarity.

Example

A detailed step by step instructions guide for going from the sequencing data to a file containing the sequences clustered by clones and with information to create isotype lineages and identify mutations is presented in Appendix A. Appendix A contains the detailed steps and instructions to set up the software packages within a computing system (here, we used Longleaf, a Linux based computing system) to go from an input ".fa" file containing sequencing data to an output ".tab" file. Each file of sequencing data contains data from a single individual and the example in Appendix A utilizes paired reads from the "JLDO" individual's B cell sequence data (file names: JLDO_S164_L001_R1_001.fastq.gz and JLDO_S164_L001_R2_001.fastq.gz). A brief overview of each package, its usage, and its purpose is presented here, and a snapshot of each corresponding output file can be found in Appendix B.

Before analysis, some steps must be taken to prepare the data. The first step is to merge the two paired ends of sequences together, which can be done using the FLASH tool. A match of at least 90% was targeted between the forward and reverse direction sequences to be combined.

The sequence file is a ".fastq" file and is then converted to a ".fa" file before proceeding onto the analysis portion.

The first analysis step is to input the sequencing data through the NCBI IgBLAST tool. This tool compares the input sequence with known germline V and J genes for the species of interest, giving an output summary of matching V, D, and J genes[13]. The JLDO_blast.sh file contains the code to run the sequencing data through the IgBLAST tool via the computing cluster. A snapshot of the output file is shown in Figure 4, which contains, for each input sequence, the best matched V(D)J germline gene as well as summary details regarding the quality of the match for regions of the sequence based on various factors (i.e. length, mismatches, gap, etc.) [13]. An overall percent identity for each portion of the sequence is also reported[13].

The next step involves the Change-O suite which contains multiple packages designed to perform a variety of analysis tasks on V(D)J sequence output, including creating clonal clusters and reconstructing clonal lineages[14]. To create a ".tab" file with information about alignment functionality, the "MakeDb" command is used to output a file of characteristics of the V and J sequences, such as start sequence, length, and germline sequences[15]. A snapshot of some of the columns contained in this output file is shown in Figure 5. Then the "DefineClones" command was used to classify the immunoglobulin sequences into clonal groups[16]. The JLDO_Clones.sh file contains the code to run the sequences through the computing cluster. This command partitions the sequencing data based on a shared V gene, J gene, and junction region length[16]. Some of the columns from the output file are shown in Figure 6, with the most notable being the newly added "CLONE" column. Three parameters were specified in the code of this command to classify sequences into clonal groups. The first parameter (-model) specifies the substitution

model to be used to calculate distances between sequences. The Hamming model was used here, which is the absolute number of differences between two sequences[9]. Then, distances were normalized by length using the normalization of distances (-norm) parameter. Finally, a default distance of 0.14 was used as the distance threshold for clonal grouping (-dist). The last command used is the "CreateGermlines" command, which adds a few more columns to the output file that will help in creating the isotype lineage and identifying mutations at various stages of B cell development[17]. A sample of this output file is shown in Figure 7, with the newly added columns added to the end of the file.

The final file obtained from processing the sequencing data with various R packages is a ".tab" file containing all relevant information to create a lineage plot, with details about both the sample and germline sequences. The information in this file can then be used in a R program (found in Appendix C), to first match the sample sequence with a list of known sequences and then create lineage trees. These known sequences corresponded to B cells identified using certain techniques, and such B cells are well documented, since we know their light chain sequences. Given each of the known B cell sequences, we identified BCR sequences from the study sample (here, data from the JLDO sample) that are similar to the designated BCR sequence. Similarity here means that the BCR sequences would likely be in the same cluster. We then conducted lineage analysis on these sequences identified, to generate lineage trees. The lineage trees help understand the timeline for differentiation and can aid in identifying mutations and isotype switching occurring during the development process[17].

Results

This analysis was conducted with data from the "JLDO" individual, which resulted in four clones that matched with the known sequences. Of these, only cluster two and three

contained a significant number of BCR sequences and a variety of isotypes that could produce a lineage tree which would contain enough data for fruitful lineage analysis. The R program helps identify the number of sequences contained within each clone, and for each clone, a lineage tree was created. Figures 2 and 3 below show lineage trees for a clone within JLDO, matching designated sequence 2 and 3, respectively.

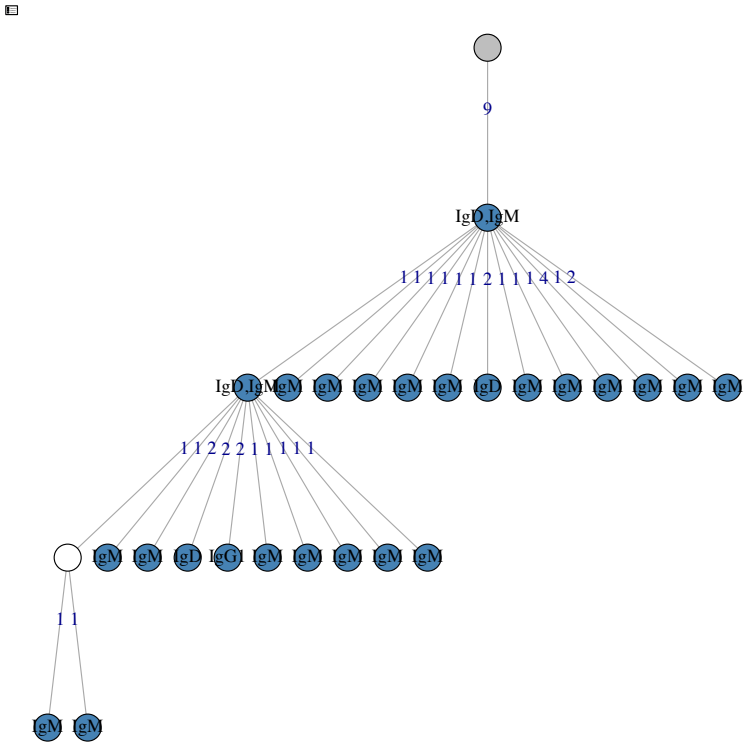


Figure 2. Lineage Tree for JLDO match sequence 2, clone number 1346.

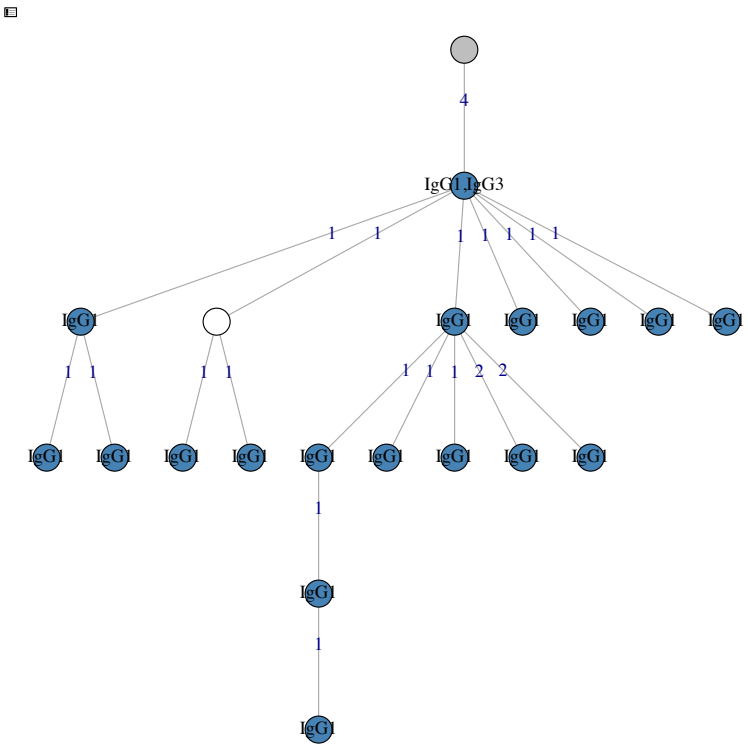


Figure 3. Lineage Tree for JLDO match sequence 3, clone number 2383.

Discussion

The lineage tree for Figure 2 contains two isotypes, IgD and IgM, which are present at multiple levels of differentiation. The multiple isotypes present at a single level can be considered clones, since they are genetically similar cells, and only one of these further differentiated. At the third differentiation step, there are IgD, IgM, and IgG1 isotypes. The presence of IgG1 isotype, despite the precursor cells only displaying IgD and IgM isotype is evidence for isotype switching. This switch could either be the result of IgD to IgG1 or IgM to IgG1. In further differentiation steps of this lineage tree, only IgM is present.

The lineage tree for Figure 3 contains two forms of the IgG isotype, labeled as IgG1 and IgG3. The figure shows that only a few different clones of IgG1 further differentiated. In each step in Figure 3, the IgG1 isotype is present and this particular tree does not show evidence of isotype switching occurring. An important difference to note between these two lineage trees is

that in Figure 2, the evidence for isotype switching occurred at the third differentiation level, while in Figure 3, by the second differentiation level, there was only IgM isotype present. This highlights how although these lineage trees are from B cells of the same individual, cells that are clonally related have a similar timeline within the same group.

While only one lineage tree per designated sequence is created here, multiple can be created per designated sequence, depending on the number of matched BCR sequences (from study sample) to the designated sequence. These lineage trees give insight into the relationship of various isotypes within an individual's B cells. If such analyses are conducted for multiple individuals and the lineage trees and results are aggregated, then patterns can be understood regarding the order of formation of these isotypes. Additionally, mutations that resulted during this isotype formation can also be discerned. Finally, this methodology points to an important characteristic of immunology with respect to B cells, called isotype switching. It is the process where a B cell switches from producing one isotype to another[9]. This analysis technique allows for the lineage trees to be subjected to further analysis to extract patterns to understand what events result in isotype switching and similarities across different types of responses.

The method of bulk sequencing of heavy and light chains is one of many methods that have been used in literature to analyze B cell data. However, combined with our data analysis approach, this method has some advantages over other commonly presented methods, namely that it is a high throughput analysis method that allows for many clones and their variants to be studied at once[3]. In this specific project we have focused on a single individual's B cell data, however, the workflow and programming that has been developed and tested allows for expansion in the future.

Other methods to analyze the heavy and light chains of an antibody that are of particular interest include hybridoma panels, antibody phage display, and single cell cloning[3]. Hybridoma panels are a technique involving white blood cells fused to myeloma cells and then gene rearrangements can be identifying and sorted according to clonal relatedness[3]. The antibody phage display technique involves using cell and molecular biology techniques to insert V genes from antibody Heavy and Light chains into phages and propagating to discover patterns of specific antibody and antigen binding[3]. Lastly, the single cell cloning method involves extracting and sequencing RNA from single cells to identify V gene sequences from Heavy and Light chains, and to express these antibody genes to test their specificity[3]. All of these methods are generally low throughput and more time intensive than the method we investigated, bulk sequencing of heavy and light chains. Interestingly, the single cell cloning method does have the potential to be applied with high throughput sequencing methods, and would be a new area of interest to further develop analysis methodologies[3].

In future experiments, we hope to continue to assess other methods for this analysis, as well as combine high throughput techniques with biologically rooted analysis methods. This immune sequencing data, organized in the correct fashion, can help further the understanding of B cell clones in mediating immune responses through phenomenon of isotype switching and somatic hypermutation.

Bibliography

1. Cano RLE, Lopera HDE. Introduction to T and B lymphocytes. In: Anaya JM, Shoenfeld Y, Rojas-Villarraga A, et al., editors. Autoimmunity: From Bench to Bedside [Internet]. Bogota (Colombia): El Rosario University Press; 2013 Jul 18. Chapter 5. Available from: https://www.ncbi.nlm.nih.gov/books/NBK459471/
2. Alberts B, Johnson A, Lewis J, et al. Molecular Biology of the Cell. 4th edition. New York: Garland Science; 2002. B Cells and Antibodies. Available from: https://www.ncbi.nlm.nih.gov/books/NBK26884/
3. Hershberg U, Prak ETL. The analysis of clonal expansions in normal and autoimmune B cell repertoires. Philosophical Transactions of the Royal Society B: Biological Sciences. 2015;370(1676):20140239. doi:10.1098/rstb.2014.0239
4. Chen Z, Collins AM, Wang Y, Gaëta BA. Clustering-based identification of clonally-related immunoglobulin gene sequence sets. Immunome Research. 2010;6(Suppl 1). doi:10.1186/1745-7580-6-s1-s4
5. Stavnezer J, Schrader CE. IgH Chain Class Switch Recombination: Mechanism and Regulation. The Journal of Immunology. 2014 [accessed 2020 Apr 7];193(11):5370–5378. doi:10.4049/jimmunol.1401849
6. Ramírez J, Lukin K, Hagman J. From hematopoietic progenitors to B cells: mechanisms of lineage restriction and commitment. Current Opinion in Immunology. 2010;22(2):177–184. doi:10.1016/j.coi.2010.02.003
7. Roth DB. V(D)J Recombination: Mechanism, Errors, and Fidelity. Microbiology Spectrum. 2014;2(6). doi:10.1128/microbiolspec.mdna3-0041-2014

8. Rock EP, Sibbald PR, Davis MM, Chien YH. CDR3 length in antigen-specific immune receptors. The Journal of Experimental Medicine. 1994;179(1):323–328. doi:10.1084/jem.179.1.323
9. Looney TJ, Lee JY, Roskin KM, Hoh RA, King J, Glanville J, Liu Y, Pham TD, Dekker CL, Davis MM, Boyd SD. Human B-cell isotype switching origins of IgE. Journal of Allergy and Clinical Immunology. 2016;137(2):579-586.e7. doi: 10.1016/j.jaci.2015.07.014.
10. Felsenstein J. Distance Methods for Inferring Phylogenies: A Justification. Evolution. 1984 [accessed 2020 Mar 19];38(1):16. doi:10.2307/2408542
11. Kepler TB. Reconstructing a B-cell clonal lineage. I. Statistical inference of unobserved ancestors. F1000Research. 2013 [accessed 2020 Mar 19];2:103. doi:10.12688/f1000research.2-103.v1
12. Gupta NT, Adams KD, Briggs AW, Timberlake SC, Vigneault F, Kleinstein SH. Hierarchical Clustering Can Identify B Cell Clones with High Confidence in Ig Repertoire Sequencing Data. The Journal of Immunology. 2017 [accessed 2020 Mar 19];198(6):2489–2499. doi:10.4049/jimmunol.1601850
13. Ye J, Ma N, Madden TL, Ostell JM. IgBLAST: an immunoglobulin variable domain sequence analysis tool. Nucleic Acids Research. 2013;41(W1). doi:10.1093/nar/gkt382
14. Gupta NT, Heiden JAV, Uduman M, Gadala-Maria D, Yaari G, Kleinstein SH. Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. Bioinformatics. 2015;31(20):3356–3358. doi:10.1093/bioinformatics/btv359
15. MakeDb. Change-O. 2018 [accessed 2020 Mar 16]. https://changeo.readthedocs.io/en/stable/tools/MakeDb.html

16. DefineClones. Change-O. 2018 [accessed 2020 Mar 16]. https://changeo.readthedocs.io/en/stable/tools/DefineClones.html

17. Reconstructing germline sequences. Change-O. 2018 [accessed 2020 Mar 16]. https://changeo.readthedocs.io/en/stable/examples/germlines.html

Appendix A: Lineage Analysis Step by Step Guide for JLDO Files

Note: Anything in red should be typed exactly as is.

1. ***Set up IgBLAST standalone tool (on Longleaf)***
    a. Make folder "IgBLAST" on linux
    b. Download ncbi-igblast-1.14.0-x64-linux.tar.gz from https://www.ncbi.nlm.nih.gov/igblast/
        i. Move this file to the new folder "IgBLAST"
    c. cd IgBLAST
    d. tar xzf ncbi-igblast-1.14.0-x64-linux.tar.gz
    e. Use IgBLAST to setup
        i. cd ncbi-igblast-1.14.0
        ii. go to http://www.imgt.org/vquest/refseqh.html download IMGT_V-Quest_reference_directory.zip
            1. from the .zip file, download IGHV.fasta, IGHD.fasta, and IGHJ.fasta from homo sapiens folder
            2. create new directory "database" under the ncbi-igblast-1.14.0 directory
            3. move the 3 .fasta files to the "database" folder
        iii. ./bin/edit_imgt_file.pl database/IGHV.fasta > database/my_IGHV
        iv. ./bin/makeblastdb -parse_seqids -dbtype nucl -in database/my_IGHV
        v. ./bin/edit_imgt_file.pl database/IGHD.fasta > database/my_IGHD
        vi. ./bin/makeblastdb -parse_seqids -dbtype nucl -in database/my_IGHD
        vii. ./bin/edit_imgt_file.pl database/IGHJ.fasta > database/my_IGHJ

viii. ./bin/makeblastdb -parse_seqids -dbtype nucl -in database/my_IGHJ

f. To test out tool, can input myseq.fa file and this will output myseq.fmt7, which should be identical to the results given by the IgBLAST online tool

i. Move myseq.fa file to IgBLAST/ncbi-igblast-1.14.0 directory

ii. ./bin/igblastn \

iii. -germline_db_V database/my_IGHV \

iv. -germline_db_D database/my_IGHD\

v. -germline_db_J database/my_IGHJ \

vi. -auxiliary_data optional_file/human_gl.aux \

vii. -domain_system imgt -ig_seqtype Ig -organism human \

viii. -outfmt '7 std qseq sseq btop' \

ix. -query myseq.fa \

x. -out myseq.fmt7

2. Using FLASH to merge two paired end sequence files - m*erge R1 and R2 data sets*

a. Use FLASH to merge R1 and R2 fastq.gz files

b. Install FLASH on linux (longleaf)

i. tar xzf FLASH-1.2.11.tar.gz

ii. cd FLASH-1.2.11

iii. make

iv. cp flash /nas/longleaf/home/onyen/.local/bin

v. echo $PATH

c. Use FLASH to combine JLDO fastq.qz files

i. flash JLDO_S164_L001_R1_001.fastq.gz JLDO_S164_L001_R2_001.fastq.gz -m 25 -x 0.25 -r 300 -f 500 -s 50 -o out1

d. Convert .fastq file to .fa file

i. paired reads in out1.extendedFraqs.fastq

1. sed -n '1~4s/^@/>/p;2~4p' out1.extendedFrags.fastq > JLDO1.fa
2. cp JLDO1.fa /nas/longleaf/home/onyen/IgBLAST/ncbi-igblast-1.14.0/JLDO.fa

3. Use IgBLAST tool to obtain .fmt7 file from .fa file with IgBLAST results

a. Run the IgBLAST tool with JLDO.fa

i. Upload the JLDO_blast.sh file under ncbi-igblast-1.14.0 directory (using CyberDuck*)

ii. chmod +x JLDO_blast.sh**

iii. sbatch JLDO_blast.sh (*note:* this submits the job to the system)

iv. *note:* can use squeue –u onyen to check the log of jobs submitted to system

4. Install Change-O tool on Linux

a. module load python

b. python3 –m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose

c. pip install biopython --user

d. Download presto-0.5.13.tar.gz from https://pypi.org/project/presto/#files

i. Move the downloaded file (presto-0.5.13.tar.gz) to IgBLAST/ncbi-igblast-1.14.0 (using CyberDuck)

e. pip3 install presto --user

f. pip3 install presto-0.5.13.tar.gz --user

g. Download changeo-0.4.6.tar.gz from https://pypi.org/project/changeo/#files

i. Move the downloaded file (changeo-0.4.6.tar.gz) to IgBLAST/ncbi-igblast-1.14.0 (using CyberDuck)

h. pip3 install changeo-0.4.6.tar.gz --user

5. ***Use Change-O command to get .tab file (from .fmt7 file obtain "_db-pass.tab" file)***

a. module load python

b. sbatch –p general –N 1 –n 1 --mem=20g –t 02-00:00:00 --wrap="MakeDb.py igblast –i JLDO1.fmt7 –s JLDO.fa –r database/IGH[VDJ].fasta --extended"

6. ***Use .tab file to define clones/lineage (obtain "_igh_clone-pass.tab" file)***

a. Upload the JLDO_clones.sh file under ncbi-igblast-1.14.0 directory (using CyberDuck

b. Use JLDO_Clones.sh file

i. chmod +x JLDO_Clones.sh***

ii. sbatch JLDO_Clones.sh (*note:* this submits the job to the system)

7. ***Use .tab file to reconstruct the germline V(D)J sequence (obtain "_igh_clone-pass_germ-pass.tab" file)***

a. module load python

b. CreateGermlines.py –d JLDO1_igh_clone.pass.tab –g dmask --cloned –r database/IGH[VDJ].fasta

***For more info on CyberDuck:** This helps transfer downloaded files/data into the research computing cluster. See FAQ #6: https://its.unc.edu/research-computing/techdocs/longleaf-frequently-asked-questions-faqs/

****Contents of the JLDO_blast.sh file:**

```
#!/bin/bash

#SBATCH -p general
#SBATCH -N 1
#SBATCH -n 16
#SBATCH --mem=100g
#SBATCH -t 02-00:00:00

./bin/igblastn -germline_db_V database/my_IGHV -germline_db_D database/my_IGHD -germline_db_J database/my_IGHJ -auxiliary_data optional_file/human_gl.aux -domain_system imgt -ig_seqtype Ig -organism human -outfmt '7 std qseq sseq btop' -num_threads 16 -query JLDO.fa  -out JLDO1.fmt7
```

*****Contents of the JLDO_Clones.sh file:**

```
#!/bin/bash

#SBATCH -p general
#SBATCH -N 1
#SBATCH -n 16
#SBATCH --mem=100g
```

#SBATCH -t 02-00:00:00

module load python

DefineClones.py -d JLDO1_db-pass.tab --model ham --norm len --dist 0.14 --outname JLDO1_igh

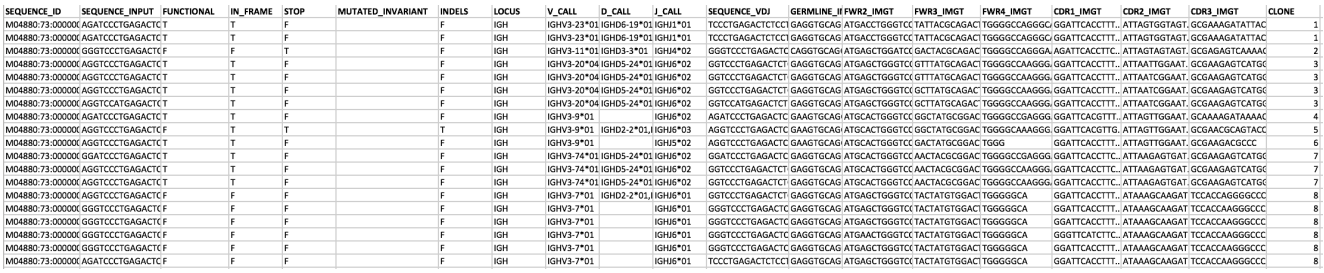
Appendix B: Snapshots of Output Files from Lineage Analysis Steps

```
# IGBLASTN
# Query: M04880:73:000000000-CL9DR:1:1101:21396:2879 1:N:0:ATCTCAGG+AAGGCTAT
# Database: database/my_IGHV database/my_IGHD database/my_IGHJ
# Domain classification requested: imgt

# V-(D)-J rearrangement summary for query sequence (Top V gene match, Top D gene match, Top J gene match, Chain type, stop codon, V-J frame, Productive, Strand).  Multiple equivalent top matches, if
present, are separated by a comma.
IGHV3-23*01,IGHV3-23*02,IGHV3-23*03     N/A     N/A     VH      No      N/A     N/A     +

# V-(D)-J junction details based on top germline gene matches (V end, V-D junction, D region, D-J junction, J start).  Note that possible overlapping nucleotides at VDJ junction (i.e, nucleotides that
could be assigned to either rearranging gene) are indicated in parentheses (i.e., (TACT)) but are not included under the V, D, or J gene itself
CTGGG   N/A     N/A     N/A     N/A

# Alignment summary between query and top germline V gene hit (from, to, length, matches, mismatches, gaps, percent identity)
FR1-IMGT        1       30      30      30      0       0       100
CDR1-IMGT       31      54      24      23      1       0       95.8
FR2-IMGT        55      64      10      8       2       0       80
Total   N/A     N/A     64      61      3       0       95.3

# Hit table (the first field indicates the chain type of the hit)
# Fields: query id, subject id, % identity, alignment length, mismatches, gap opens, gaps, q. start, q. end, s. start, s. end, evalue, bit score, query seq, subject seq, BTOP
# 3 hits found
V       M04880:73:000000000-CL9DR:1:1101:21396:2879     IGHV3-23*01     95.312  64      3       0       0       1       64      46      109     8.46e-22        92.2
GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGAGTCACCTTTAGCAGCTATGCCATGCCCTGGG       GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGATTCACCTTTAGCAGCTATGCCATGAGCTGGG       33GT23CACG5
V       M04880:73:000000000-CL9DR:1:1101:21396:2879     IGHV3-23*02     95.312  64      3       0       0       1       64      46      109     8.46e-22        92.2
GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGAGTCACCTTTAGCAGCTATGCCATGCCCTGGG       GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGATTCACCTTTAGCAGCTATGCCATGAGCTGGG       33GT23CACG5
V       M04880:73:000000000-CL9DR:1:1101:21396:2879     IGHV3-23*03     95.312  64      3       0       0       1       64      46      109     8.46e-22        92.2
GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGAGTCACCTTTAGCAGCTATGCCATGCCCTGGG       GGGTCCCTGAGACTCTCCTGTGCAGCCTCTGGATTCACCTTTAGCAGCTATGCCATGAGCTGGG       33GT23CACG5
```

Figure 4. Output from JLDO1.fmt7 file containing data from IgBLAST tool for a single sequence. The output shows the parts of the sequence that match with the V(D)J germline sequence.

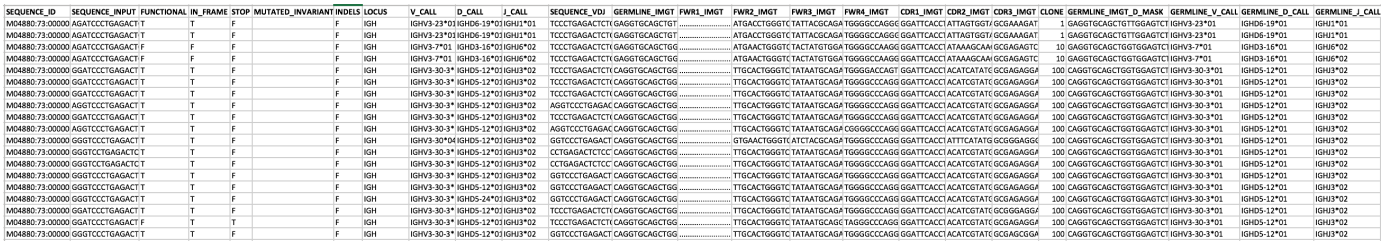
SEQUENCE_ID	SEQUENCE_INPUT	FUNCTIONAL	IN_FRAME	STOP	MUTATED_INVARIANT	INDELS	LOCUS	V_CALL	D_CALL	J_CALL	SEQUENCE_VDJ	SEQUENCE_IMGT	V_SEQ_START	V_SEQ_LENGTH	V_GERM_START_VDJ	V_GERM_LENGTH_VDJ	V_GERM_START_IMGT	V_GERM_LENGTH_IMGT	NP1_LENGTH	D_SEQ_START	D
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*0(	IGHD3-16*0	IGHJ6*02	GGGTCCCTGAGACT	.....	1	250	46	250	1	319	9	260	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*0(	IGHD5-18*0	IGHJ3*02	GGGTCCCTGAGACT	.....	1	251	46	251	1	320	5	257	
M04880:73:0000	GGGTCCTGAGACTCT	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*01,IG	CCTGAGACTCTACT	.....	5	242	51	242	1	316	8	255	
M04880:73:0000	GGATCCTGAGACTCT	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	CCTGAGACTCTCCT	.....	5	242	51	242	1	316	8	255	
M04880:73:0000	AGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-48*0(	IGHD3-16*0	IGHJ6*02	TCCCTGAGACTCTC	.....	4	247	49	247	1	319	9	260	
M04880:73:0000	AGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	TCCCTGAGACTCTC	.....	4	244	49	244	1	316	8	256	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*0(	IGHD3-16*0	IGHJ6*02	GGGTCCCTGAGACT	.....	1	250	46	250	1	319	9	260	
M04880:73:0000	AGGTCCCTGAGACTC	F	F	F		F	IGH	IGHV3-11*0(	IGHD2-21*0	IGHJ4*02	GGCCGATTCACCAT	.....	44	99	196	99	1	318	7	150	
M04880:73:0000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*1(	IGHD2-2*01,	IGHJ6*03	TCCCTGAGACTCTC	.....	4	245	49	245	1	317	3	252	
M04880:73:0000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*1(	IGHD2-2*01,	IGHJ6*03	TCCCTGAGACTCTC	.....	4	245	49	245	1	317	3	252	
M04880:73:0000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30*0(	IGHD1-26*0	IGHJ6*02	TCCCTGAGACTCTC	.....	4	248	49	248	1	320	5	257	
M04880:73:0000	AGTCCCTGAGACTCT	T	T	F		F	IGH	IGHV3-30*0(	IGHD3-16*0	IGHJ6*02	GTCCCTGAGACTCT	.....	2	248	48	248	1	319	9	259	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	GGGTCCCTGAGACT	.....	1	247	46	247	1	316	8	256	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-15*0(	IGHD1-14*0	IGHJ4*02	GGGTCCCTGAGACT	.....	1	257	46	257	1	320	6	264	
M04880:73:0000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-48*0(	IGHD2-15*0	IGHJ6*02	GGATCCCTGAGACT	.....	1	251	46	251	1	320	4	256	
M04880:73:0000	AGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-7*01	IGHD1-26*0	IGHJ6*02	TCCCTGAGACTCTC	.....	4	248	49	248	1	320	5	257	
M04880:73:0000	AGGTCCCTGAGACTC	F	T	T		T	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	GGTCCCTGAGACTC	.....	2	245	47	246	1	316	8	255	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD3-16*0	IGHJ6*02	GGGTCCCTGAGACT	.....	1	250	46	250	1	319	9	260	
M04880:73:0000	AGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-74*0(	IGHD5-18*0	IGHJ4*02	GGTCCCTGAGACTC	.....	2	249	47	249	1	319	13	264	
M04880:73:0000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD2-21*0	IGHJ4*02,IG	GGATCCCTGAGACT	.....	1	251	46	251	1	320	5	257	
M04880:73:0000	AGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	GGTCCCTGAGACTC	.....	2	246	47	246	1	316	8	256	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	GGGTCCCTGAGACT	.....	1	247	46	247	1	316	8	256	
M04880:73:0000	GGGTCCCTGAGACTC	F	T	T		F	IGH	IGHV3-7*01	IGHD1-26*0	IGHJ6*02	GGGTCCCTGAGACT	.....	1	251	46	251	1	320	5	257	
M04880:73:0000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*0(	IGHD1-7*01,	IGHJ3*02	GGGTCCCTGAGACT	.....	1	247	46	247	1	316	8	256	
M04880:73:0000	AGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-66*0(	IGHD3-10*0	IGHJ3*02	GGTCCCTGAGACTC	.....	2	243	47	243	1	316	3	248	

Figure 5. Output from JLDO1_db-pass.tab containing characteristics of the V and J sequences.



SEQUENCE_ID	SEQUENCE_INPUT	FUNCTIONAL	IN_FRAME	STOP	MUTATED_INVARIANT	INDELS	LOCUS	V_CALL	D_CALL	J_CALL	SEQUENCE_VDJ	GERMLINE_IF	FWR2_IMGT	FWR3_IMGT	FWR4_IMGT	CDR1_IMGT	CDR2_IMGT	CDR3_IMGT	CLONE
M04880:73:00000(	AGATCCCTGAGACTC	T		T	F		F	IGH	IGHV3-23*01	IGHD6-19*01	IGHJ1*01	TCCCTGAGACTCTCC	GAGGTGCAG	ATGACCTGGGTCC	TATTACGCAGAC	TGGGGCCAGGGG	GGATTCACCTTT	ATTAGTGGTAGT	GCGAAAGATATTAC	1
M04880:73:00000(	AGATCCCTGAGACTC	T		T	F		F	IGH	IGHV3-23*01	IGHD6-19*01	IGHJ1*01	TCCCTGAGACTCTCC	GAGGTGCAG	ATGACCTGGGTCC	TATTACGCAGAC	TGGGGCCAGGGG	GGATTCACCTTT	ATTAGTGGTAGT	GCGAAAGATATTAC	1
M04880:73:00000(	GGGTCCCTGAGACTC	F		F	T		F	IGH	IGHV3-11*01	IGHD3-3*01	IGHJ4*02	GGGTCCCTGAGACTC	CAGGTGCAG	ATGAGCTGGATCC	GACTACGCAGAC	TGGGGCCAGGGA	GGATTCACCTTC	ATTAGTAGTAGT	GCGAGAGTCAAAAC	2
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-20*04	IGHD5-24*01	IGHJ6*02	GGTCCCTGAGACTCT	GAGGTGCAG	ATGAGCTGGGTCC	GTTTATGCAGAC	TGGGGCCAAGGG	GGATTCACCTTT	ATTAATGGAAT	GCGAAGAGTCATGC	3
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-20*04	IGHD5-24*01	IGHJ6*02	GGTCCCTGAGACTCT	GAGGTGCAG	ATGAGCTGGGTCC	GTTTATGCAGAC	TGGGGCCAAGGG	GGATTCACCTTT	ATTAATCGGAAT	GCGAAGAGTCATGC	3
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-20*04	IGHD5-24*01	IGHJ6*02	GGTCCCTGAGACTCT	GAGGTGCAG	ATGAGCTGGGTCC	GCTTATGCAGAC	TGGGGCCAAGGG	GGATTCACCTTT	ATTAATCGGAAT	GCGAAGAGTCATGC	3
M04880:73:00000(	AGGTCCCATGAGACTC	T		T	F		F	IGH	IGHV3-20*04	IGHD5-24*01	IGHJ6*02	GGTCCATGAGACTCT	GAGGTGCAG	ATGAGCTGGGTCC	GCTTATGCAGAC	TGGGGCCAAGGG	GGATTCACCTTT	ATTAATCGGAAT	GCGAAGAGTCATGC	3
M04880:73:00000(	AGATCCCTGAGACTC	T		T	F		F	IGH	IGHV3-9*01		IGHJ6*02	AGATCCCTGAGACTC	GAAGTGCAG	ATGCACTGGGTCC	GGCTATGCGGAC	TGGGGCCGAGGG	GGATTCACGTTT	ATTAGTTGGAAT	GCAAAAGATAAAAC	4
M04880:73:00000(	AGGTCCCTGAGACTC	F		T	T		T	IGH	IGHV3-9*01	IGHD2-2*01,	IGHJ6*03	AGGTCCCTGAGACTC	GAAGTGCAG	ATGCACTGGGTCC	GGCTATGCGGAC	TGGGGCAAAGGG	GGATTCACGTTG	ATTAGTTGGAAT	GCGAACGCAGTACC	5
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-9*01		IGHJ5*02	AGGTCCCTGAGACTC	GAAGTGCAG	ATGCACTGGGTCC	GACTATGCGGAC	TGGG	GGATTCACCTTT	ATTAGTTGGAAT	GCGAAGACGCCC	6
M04880:73:00000(	GGATCCCTGAGACTC	T		T	F		F	IGH	IGHV3-74*01	IGHD5-24*01	IGHJ6*02	GGATCCCTGAGACTC	GAGGTGCAG	ATGCACTGGGTCC	AACTACGCGGAC	TGGGGCCAAGGG	GGATTCACCTTC	ATTAAGAGTGAT	GCGAAGAGTCATGC	7
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-74*01	IGHD5-24*01	IGHJ6*02	GGTCCCTGAGACTCT	GAGGTGCAG	ATGCACTGGGTCC	AACTACGCGGAC	TGGGGCCAAGGG	GGATTCACCTTC	ATTAAGAGTGAT	GCGAAGAGTCATGC	7
M04880:73:00000(	AGGTCCCTGAGACTC	T		T	F		F	IGH	IGHV3-74*01	IGHD5-24*01	IGHJ6*02	GGTCCCTGAGACTCT	GAGGTGCAG	ATGCACTGGGTCC	AACTACGCGGAC	TGGGGCCAAGGG	GGATTCACCTTC	ATTAAGAGTGAT	GCGAAGAGTCATGC	7
M04880:73:00000(	AGGTCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01	IGHD2-2*01,1	IGHJ6*01	GGTCCCTGAGACTCT	GAGGTGCAG	ATGAGCTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGATTCACCTTT	ATAAAGCAAGAT	TCCACCAGGGGCCC	8
M04880:73:00000(	GGGTCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01		IGHJ6*01	GGGTCCCTGAGACTC	GAGGTGCAG	ATGAGCTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGATTCACCTTT	ATAAAGCAAGAT	TCCACCAGGGGCCC	8
M04880:73:00000(	GGGTCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01		IGHJ6*01	GGGTCCCTGAGACTC	GAGGTGCAG	ATGAGCTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGATTCACCTTT	ATAAAGCAAGAT	TCCACCAGGGGCCC	8
M04880:73:00000(	GGGTCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01		IGHJ6*01	GGGTCCCTGAGACTC	GAGGTGCAG	ATGAACTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGGTTCATCTTC	ATAAAGCAAGAT	TCCACCAAGGGCCC	8
M04880:73:00000(	GGGTCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01		IGHJ6*01	GGGTCCCTGAGACTC	GAGGTGCAG	ATGAGCTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGATTCACCTTT	ATAAAGCAAGAT	TCCACCAAGGGCCC	8
M04880:73:00000(	AGATCCCTGAGACTC	F		F	F		F	IGH	IGHV3-7*01		IGHJ6*01	TCCCTGAGACTCTCC	GAGGTGCAG	ATGAGCTGGGTCC	TACTATGTGGAC	TGGGGGCA	GGATTCACCTTT	ATAAAGCAAGAT	TCCACCAAGGGCCC	8

Figure 6. Output from JLDO1_igh_clone-pass.tab, containing information from Figure 5 in addition to the "CLONE" column. The middle columns are omitted in this figure as they are identical to those in Figure 5.



SEQUENCE_ID	SEQUENCE_INPUT	FUNCTIONAL	IN_FRAME	STOP	MUTATED_INVARIANT	INDELS	LOCUS	V_CALL	D_CALL	J_CALL	SEQUENCE_VDJ	GERMLINE_IMGT	FWR1_IMGT	FWR2_IMGT	FWR3_IMGT	FWR4_IMGT	CDR1_IMGT	CDR2_IMGT	CDR3_IMGT	CLONE	GERMLINE_IMGT_D_MASK	GERMLINE_V_CALL	GERMLINE_D_CALL	GERMLINE_J_CALL
M04880:73:00000	AGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-23*01	IGHD6-19*01	IGHJ1*01	TCCCTGAGACTCTCC	GAGGTGCAGCTGT	.....	ATGACCTGGGTCC	TATTACGCAGAT	TGGGGCCAGGGG	GGATTCACCTAT	TAGTGGTAGT	GCGAAAGAT	1	GAGGTGCAGCTGTTGGAGTCT	IGHV3-23*01	IGHD6-19*01	IGHJ1*01
M04880:73:00000	AGATCCCTGAGACTC	T	F	F		F	IGH	IGHV3-23*01	IGHD6-19*01	IGHJ1*01	TCCCTGAGACTCTCC	GAGGTGCAGCTGT	.....	ATGACCTGGGTCC	TATTACGCAGAC	TGGGGCCAGGGG	GGATTCACCTAT	TAGTGGTAGT	GCGAAAGAT	1	GAGGTGCAGCTGTTGGAGTCT	IGHV3-23*01	IGHD6-19*01	IGHJ1*01
M04880:73:00000	AGATCCCTGAGACTC	F	F	F		F	IGH	IGHV3-7*01	IGHD3-16*01	IGHJ6*02	TCCCTGAGACTCTCC	GAGGTGCAGCTGG	.....	ATGAACTGGGTCC	TACTATGTGGA	TGGGGCCAAGG	GGATTCACCTAT	AAAGCAAG	GCGAGAGTC	10	GAGGTGCAGCTGGTGGAGTCT	IGHV3-7*01	IGHD3-16*01	IGHJ6*02
M04880:73:00000	AGATCCCTGAGACTC	F	F	F		F	IGH	IGHV3-7*01	IGHD3-16*01	IGHJ6*02	TCCCTGAGACTCTCC	GAGGTGCAGCTGG	.....	ATGAACTGGGTCC	TACTATGTGGA	TGGGGCCAAGG	GGATTCACCTAT	AAAGCAAG	GCGAGAGTC	10	GAGGTGCAGCTGGTGGAGTCT	IGHV3-7*01	IGHD3-16*01	IGHJ6*02
M04880:73:00000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	TCCCTGAGACTCTCC	GAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCAGT	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	TCCCTGAGACTCT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGATCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	TCCCTGAGACTCT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	AGGTCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	AGGTCCCTGAGAC	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGATCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	TCCCTGAGACTCT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	AGGTCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	AGGTCCCTGAGAC	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	CGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30*0	IGHD5-12*01	IGHJ3*02	GGTCCCTGAGACT	CAGGTGCAGCTGG	.....	GTGAACTGGGTC	ATCTACGCAGA	TGGGGCCCAGG	GGATTCACCT	ATTTCATAT	GCGGGAGGC	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	CCTGAGACTCTCC	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	CCTGAGACTCTCC	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	GGTCCCTGAGACT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCCTGAGACTC	T	F	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	GGTCCCTGAGACT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGATCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	TCCCTGAGACTCT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGC	TAGGGGCCAGG	GGATTCACCT	ACATCGTAT	GCGAGAGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02
M04880:73:00000	GGGTCCCTGAGACTC	T	T	F		F	IGH	IGHV3-30-3*	IGHD5-12*01	IGHJ3*02	GGTCCCTGAGACT	CAGGTGCAGCTGG	.....	TTGCACTGGGTC	TATAATGCAGA	TGGGGCCCAGG	GGATTCACCT	ACATCGTAT	GCGAGCGGA	100	CAGGTGCAGCTGGTGGAGTCT	IGHV3-30-3*01	IGHD5-12*01	IGHJ3*02

Figure 7. Output from JLDO1_igh_clone-pass_germ-pass.tab, containing information from Figure 6 in addition to the "GERMLINE_IMGT_D_MASK", "GERMLINE_V_CALL", "GERMLINE_D_CALL", and "GERMLINE_J_CALL" columns. The middle columns are omitted in this figure as they are identical to those in Figure 6.

Appendix C: R Code to Produce Lineage Plots from .tab File

```
#To match sequences in samples from "_igh_clone-pass_germ-pass.tab"
 files with a list of sequences of known mAB (FS mAB H chain V region
 sequences (VH3 and VH4))

setwd("/Users/ishanik/Desktop")

library(alakazam)
library(tigger)
library(openxlsx)
library("stringdist")
```

```
library(stringr)
library(igraph)
library(dplyr)
library(openxlsx)

rm(list = ls.str())

myref <- read.xlsx("FS mAb H chain V region sequences (VH3 and
 VH4).xlsx") # unique(substr(myref$mAb, 1,4))

 #can change depending on the header of the sample sequence
idx <- which( substr(myref$mAb, 1,4) == 'JLDO')
headname <- "JLDO1"

db <- readChangeoDb(paste0(headname, "_igh_clone-pass_germ-pass.tab")
 )

seq.select <- list()
for (j in 1:length(idx) ) ## nrow(myref) )  ## first 6 rows for JLDO,
 remaining 6 rows for GCDS
{
  jj <- idx[j]
  tmp_sub <- subset(db, nchar(CDR3_IMGT) == nchar(myref$CDR3[jj] ) )
  bb_dist <- stringdistmatrix(tmp_sub$CDR3_IMGT, myref$CDR3[jj],
 method = "hamming" )
  IDX <- (bb_dist < 0.25*nchar(myref$CDR3[jj] ) )
  H_dist <- 1 - bb_dist[IDX]/nchar(myref$CDR3[jj] )
  res_sub <- cbind(tmp_sub[IDX, ], H_dist )
  seq.select[[j]] <- res_sub
}
save(seq.select, myref, headname, idx, file = paste0(headname,
 "_CDR3_Germ_selected.RData" ) )

## get constant region ...
IgM <-
 toupper("gggagtgcatccgccccaacccttttcccccctcgtctcctgtgagaattccc")
IgD <-  toupper("gcacccaccaaggctccggatgtgttccccatcatatcaggg")
IgG1    <-
 toupper("gcctccaccaagggcccatcggtcttcccccctggcaccctcctccaagagcacctctgggg
 gcacagcggccctgggctgcctg")
IgG2    <-
 toupper("cctccaccaagggcccatcggtcttcccccctggcgccctgctccaggagcacctccgagag
 cacagccgccctgggctgcctg")
IgG3    <-
```

```
 toupper("cttccaccaagggcccatcggtcttccccctggcgccctgctccaggagcacctctggggg
 cacagcggccctgggctgcctg")
IgG4    <-
 toupper("gcttccaccaagggcccatccgtcttccccctggcgccctgctccaggagcacctccgaga
 gcacagccgccctgggctgcctg")
IgA1    <-
 toupper("gcatccccgaccagccccaaggtcttcccgctgagcctctgcagcacccagccagatggga
 acgtggtcatcgcctgcctggtccagggct")
IgA2    <-
 toupper("gcatccccgaccagccccaaggtcttcccgctgagcctcgacagcacccccccaagatggga
 acgtggtcgtcgcatgcctggtccagggct")
IgE <-  toupper("gcctccacacagagcccatccgtcttc")
IgName <- c("IgM", "IgD", "IgG1", "IgG2", "IgG3", "IgG4", "IgA1",
 "IgA2", "IgE")

load(file = paste0(headname, "_CDR3_Germ_selected.RData" ) )
load( file = "JLDO1_CDR3_Germ_selected.RData" )
for (i in  1:length(seq.select) ){
  tmp <- seq.select[[i]]
  if ( nrow(tmp) > 0) {
    refVgene <- myref$VH.gene.sequence[idx[i]]
    V_seqs_raw <- tmp$SEQUENCE_VDJ
    loc_start <- tmp$V_SEQ_START
    loc_end <- tmp$V_SEQ_START + tmp$V_SEQ_LENGTH - 1
    V_gene <- substr(V_seqs_raw, loc_start, loc_end )

    bb_dist <- stringdistmatrix(V_gene, refVgene, method = "osa" )
    tmp$V_dist <- as.vector(bb_dist)
    tmp$V_similarity <- 1 - as.vector(tmp$V_dist / nchar(refVgene))
    tmp <- tmp[order(tmp$V_dist),]

    db_sub <- tmp
    db_sub$anchar <- str_length(db_sub$SEQUENCE_INPUT)
    apos <- str_locate(db_sub$SEQUENCE_INPUT, db_sub$FWR4_IMGT)
    db_sub$CH_seq <- substr(db_sub$SEQUENCE_INPUT, apos[, 2],
 db_sub$anchar)
    db_sub$CH_loc1 <- apos[,1]
    db_sub$CH_loc2 <- apos[,2]

    bb_dist <- stringdistmatrix(db_sub$CH_seq, c(IgM, IgD, IgG1, IgG2,
 IgG3, IgG4, IgA1, IgA2, IgE) )
    c.indx <- apply(bb_dist, 1,  which.min )
    c.indx_ <- do.call(c, lapply(c.indx, (function(x) {
      if (is.null(x) | length(x) == 0) {NA} else { x }
```

```
    }))))
    c.dist <- unlist(apply(bb_dist, 1,  min ))
    db_sub$c.dist <- c.dist
    db_sub$IgName <- IgName[c.indx_]
    colnames(bb_dist) <- IgName



    write.xlsx(cbind(db_sub$IgName, db_sub, bb_dist),
 file=paste0(headname, "_Match_seq", i, ".xlsx") ) # ,
 sheetName=paste(i), append=T)
  }
  else{
    write.xlsx(tmp, file=paste0(headname, "_Match_seq", i, ".xlsx") )
 # , sheetName=paste(i), append=T)
  }
}

#can change depending on the header of the sample sequence
headname <- "JLD01"
i <- 3
myseq <- read.xlsx(paste0(headname, "_Match_seq", i, ".xlsx") )

db_sub_Clone <- subset(myseq, select = c(SEQUENCE_ID, SEQUENCE_INPUT,
 SEQUENCE_IMGT, GERMLINE_IMGT_D_MASK, CLONE, V_CALL, J_CALL,
 JUNCTION_LENGTH, IgName)  )

colnames(db_sub_Clone)[9] <- "ISOTYPE"
table(db_sub_Clone$CLONE, db_sub_Clone$ISOTYPE)

sub_db_a <- subset(db_sub_Clone, CLONE == 2383)  #REPLACE this w/
 appropriate clone number
sub_db <- sub_db_a[!duplicated(sub_db_a[c("SEQUENCE_INPUT")]),]
sub_db <- sub_db[1:40, ]
clone <- makeChangeoClone(sub_db, text_fields=c("ISOTYPE"),
 pad_end=TRUE)

#Download the following from:
 http://evolution.genetics.washington.edu/phylip/getme-new1.html
dnapars_exec <- "/Users/ishanik/Desktop/phylip-3.695/exe/dnapars"
graph <- buildPhylipLineage(clone, dnapars_exec, rm_temp=FALSE)

#output sequences
my.out <- cbind(c(1:length(V(graph))),
```

```
 data.frame(SEQUENCE_ID=V(graph)$name, ISOTYPE=V(graph)$ISOTYPE))
my.out.1 <- merge(x = my.out, y = myseq, by = "SEQUENCE_ID", all.x =
 TRUE)
write.csv(my.out.1, file = paste0(headname, "_Tree_seq", i, ".csv") )

#Customized Plot
#With sequence labels
pdf(file = paste0( paste0(headname, "_Tree_seq", i, ".pdf")  ) )
V(graph)$color <- "steelblue"
V(graph)$color[V(graph)$name == "Germline"] <- "grey"
V(graph)$color[grepl("Inferred", V(graph)$name)] <- "white"
V(graph)$label <- 1:length(V(graph)$label)
par(mar=c(0, 0, 0, 0) + 0.1)

#Plot graph
plot(graph, layout=layout_as_tree, edge.arrow.mode=0,
 vertex.frame.color="black",
     vertex.label.color="black", vertex.size=8)

#Add legend
legend("topleft", c("Germline", "Inferred", "Sample"),
       fill=c("black", "white", "steelblue"), cex=0.75)

#With sequence Isotypes
V(graph)$color <- "steelblue"
V(graph)$color[V(graph)$name == "Germline"] <- "grey"
V(graph)$color[grepl("Inferred", V(graph)$name)] <- "white"
V(graph)$label <-  V(graph)$ISOTYPE
par(mar=c(0, 0, 0, 0) + 0.1)

# Plot graph
plot(graph, layout=layout_as_tree, edge.arrow.mode=0,
 vertex.frame.color="black",
     vertex.label.color="black", vertex.size=8)

# Add legend
legend("topleft", c("Germline", "Inferred", "Sample"),
       fill=c("black", "white", "steelblue"), cex=0.75)
dev.off()
```